基于 ARM920T 的嵌入式图像处理平台搭建及应用①

姚宇乐, 陈　强, 张九卫, 徐晨峰

(江西理工大学 电气工程与自动化学院, 赣州 341000)

摘　要: 研究了在 Linux OS 上交叉编译 OpenCV 和 Qt 后将其移植到嵌入式 Linux 操作系统的方法. 搭建了以 ARM920T 为核心的嵌入式图像处理平台, 能够实现复杂图像算法的处理. 在此平台上实现了基于经典算法的条形码识别功能. 描述了条形码定位和识别过程及算法. 实验结果验证了本文方法的有效性.

关键词: OpenCV; 嵌入式; Qt; 条形码识别

Construction and Application of Embedded Image Processing System Based on ARM920T

YAO Yu-Le, CHEN Qiang, ZHANG Jiu-Wei, XU Chen-Feng

(School of Electrical Engineering and Automation, Jiangxi University of Science and Technology, Ganzhou 341000, China)

Abstract: The article research on a method that corss compiling OpenCV and Qt on Linux OS and porting it to Linux operating system. It builds up an Embedded Image Processing System with ARM920T which is a core that can achieve complicated image processing. This platform makes classic algorithm barcode recognition come true. The article describes barcode position recognition and algorithm. The experimental result verifies the availability of the method in this article.

Key words: OpenCV; embedded; Qt; barcode recognition

相对于传统的图像处理技术, 嵌入式图像处理具有实时处理、便携性好、代码固件化、软件短小精悍、成本合理、低功耗、抗干扰性好七大优点, 但也存在硬件方面的处理器处理速度不够快、主频高低的限制, 软件方面的设计难度大、需要设计人员专业性很强等问题. 针对上述内容, 本文搭建了一种基于 ARM9 的嵌入式图像处理平台, 使用 OpenCV 的函数库来实现图像处理和计算机视觉方面的通用算法, 降低开发难度, 缩短软件开发周期[1].

1　嵌入式图像处理硬件平台搭建

本文所搭建的系统是由嵌入式硬件平台, 嵌入式操作系统, 嵌入式软件平台以及实现的应用程序组成. 本文所研究的嵌入式图像处理系统的组成结构图如图 1 所示.

本文主要以友善之臂 MINI2440 的核心板作为硬件平台, 摄像头采用中星微 ZC301P, 通过 USB 接口与核心板连接, 结合显示模块、普通输出输入模块, 为图像处理提供摄像头采集、实时处理和显示支持. MINI2440 核心板所使用的内核为 ARM920T[2], 该内核使用的是五级流水线, 主频可达 300MHz, 具有指令和数据 Cache, 支持协处理器、片上调试和 MMU 等技术, 并采用哈佛体系结构[3], 可以使得数据的吞吐率提高一倍.

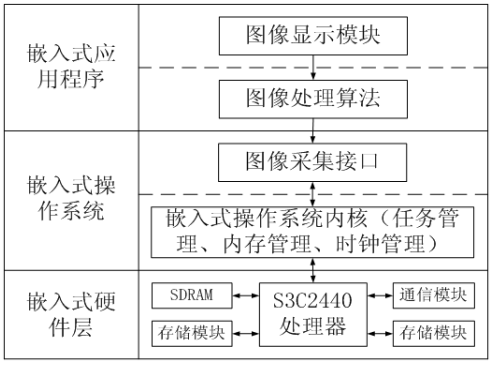

图 1　嵌入式图像处理系统结构图

① 收稿时间:2015-10-28;收到修改稿时间:2015-12-10　[doi: 10.15888/j.cnki.csa.005250]

2 嵌入式图像处理软件平台搭建

软件平台的搭建主要是 OpenCV 和 Qt 的移植. OpenCV 提供了非常丰富的帧提取函数和视觉处理算法, 开发者可以在其视频开发项目中直接调用进行算法移植, 并添加自己编写的程序, 即可完成复杂庞大的开发任务, 达到事半功倍的效果. OpenCV 的 HighGUI 模块中的函数是基于 GTK+的, 而将图像显示在 GTK+窗体上的移植相当繁琐, 容易出错, 并且在嵌入式图像处理开发平台中, 应用程序的开发与 GTK+相关性不大, 因此本文提出采用 Qt 编写图像接口部分. Qt 即图形用户界面应用程序开发框架, 它提供给应用程序开发者建立艺术级的图形用户界面所需的所有功能.

2.1 图形界面设计工具 Qt 移植

Qt 是一种图形界面设计工具, 同 XWindows 上的 Motif, Openwin, GTK 以及 Windows 平台上的 MFC, OWL, VCL, ATL 等是同类型的设计工具. 本文选用 Qt-4.8.5, 其在嵌入式 Linux 的移植主要步骤如下:

第一步: 解压 Qt[4]: tar -zxvf qt-everywhere-opensource-src-4.8.5.tar.gz, 并使用 mv qt-everywhere-opensource-src-4.8.5 qt-4.8.5-arm 将解压后的文件夹重命名为 qt-4.8.5-arm.

第二步: 在终端中进入解压之后的文件目录开始配置. 在终端中输入命令:

./configure -prefix /usr/local/Trolltech/Qt-4.8.5-arm -release -shared -fast -opensource -no-3dnow -no-openssl -no-libmng -no-opengl -no-qvfb -no-glib -no-phonon -nomake examples -nomake tools -nomake docs -nomake demos -qt-sql-sqlite -qt-libjpeg -qt-zlib -qt-libpng -qt-mouse-tslib -xplatform qws/linux-arm-g++ -embedded arm -little-endian -depths 16 -confirm-license -I/opt/tslib/include -L/opt/tslib/lib

其中-no-xxx 表示配置 Qt 不支持 xxx, 而-qt-xxx 表示 Qt 支持 xxx, 这里制定了 Qt 的安装路径 /usr/local/Trolltech/Qt-4.8.5-arm, Qt 目标架构 arm, 编译平台 linux-arm-g++.

第三步:在端口输入 make 和 make install 安装完成后, 将经过 arm-linux-gcc 编译得到的 lib 包通过网络文件系统下载到嵌入式平台的根文件系统下. 并修改 etc/profile 文件, 添加 Qt 库的环境变量.

2.2 计算机视觉库 OpenCV 移植

本文所移植的计算机视觉库为 OpenCV-2.3.1, 交叉编译器为 arm-linux-gcc-4.4.3, 主要移植分为如下几步:

第一步: 下载并安装 CMake. CMake 是一个跨平台的编译工具, 能够输出各种 makefile 或者 project 文件, 可以编译源代码、制作程式库、产生适配器、还可以用任意的顺序构建执行档. 而 OpenCV-2.0 之后的版本, 必须使用 CMake 创建 Makefike, 本文使用 cmake-2.8.4-Linux-i386, 解压到目录/usr/local/cmake-2.8.4-Linux-i386, 然后在环境变量 PATH 中增加 /usr/local/cmake-2.8.4-Linux-i386/bin.

第二步: 下载并解压 OpenCV[5], 进入解压之后的文件目录, 在终端中输入 cmake-gui 进行配置. 选择源代码目录/usr/local/OpenCV-2.3.1, 选择 Build 目录 /usr/local/opencv-arm/, 点击 Next, 分别选择 C 和 C++ 编译器, 修改安装目录 CMAKE_INSTALL_PREFIX. 由于没有安装 tiff 图像的支持, 因此去掉 WIFF 选项, 以免使编译产生错误.

进入 cmake 所选的 Build 目录, 在终端中输入 make 和 make install 完成安装.

第三步: 进入安装目录, 将该目录下编译链接得到的三个文件: include、lib、share 通过网络文件系统下载到嵌入式平台上, 在实际使用该图像处理平台时, 通过网络文件系统将在 PC 机上交叉编译完成的二进制可执行文件下载到嵌入式平台上后, 需要在嵌入式平台上进行动态库路径的设置. 在终端中输入命令:export LD_LIBRARY_PATH=[相应 lib 库的路径].

将安装目录下的 include 中的头文件复制到 PC 机上交叉编译器的 incude 目录下, 将安装目录下的 lib 中动态文件库全部复制到 PC 机上交叉编译器的 lib 目录下. 可以使得在 PC 上编译 OpenCV 应用程序时有 OpenCV 库文件的支持.

3 嵌入式平台的条形码处理

图像处理系统在嵌入式平台上运行对在其上运行的系统有着许多要求, 如:功耗、可靠性、体积、处理速度、功能、成本等方面, 这些要求对设计者来说是一大挑战.

本文主要以识别 EAN-13 商品条形码为例来检测嵌入式图像处理平台的可行性和稳定行, EAN-13 码[6]

是通过 13 个数字来标示一种商品, 典型的 EAN-13 码结构主要有左侧空白区、起始符、左侧数据符、中间分隔符、右侧分隔符、检验符、终止符、右侧空白区以及供人识别字符组成. 如表 1 所示.

表 1　EAN-13 码结构

左侧空白区	起始符	左侧数据符	中间分隔符	右侧数据符	校验符	终止符	右侧空白符
9 个模块	3 个模块	42 个模块	5 个模块	35 个模块	7 个模块	3 个模块	9 个模块

在条形码中, 每一条形码字符由 7 个模块组成, 每个模块表达 0 或者 1, 每个字符含条或空个数各为 2, 相邻元素如果相同, 则从外观上合并为一个条或空, 并规定每个字符在外观上包含的条或者空的个数必须各为 2 个, 所以 EAN 码是一种(7, 2)码.

4　条形码定位识别过程及算法

由于在实际摄像头拍摄过程中, 条形码所处的位置高低远近变化无常, 再加上光线的变化, 相机本身曝光时间, 拍摄时相机的角度及振动等因素, 会影响所采集图像的质量. 故在对条形码的图像进行信息识别之前, 需要对图像进行校正和补偿, 以减少识别的错误率. 因此要对图像特征进行处理. 这些工作依次为对采集到的图像进行噪声滤除, 灰度化处理, 边缘化处理, 图像二值化, 倾斜校正, 最终将得到的图像特征进行提取, 如图 2 所示.

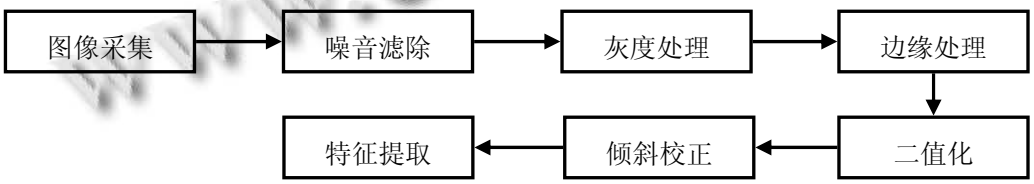

图 2　图像校正与处理流程

4.1 条形码图像的噪声过滤

本文采用均值滤波[7]抑制噪声, 即用均值来替代原图像中的各个像素值. 主要是来消除非条形码边缘的直线对条形码定位的干扰.

均值滤波的方法如下:设备噪声污染的信号为 x(t), 原始信号为 y(t), 噪声为 f(t), 即公式(1)所示.

$$x(t)=y(t)+f(t) \tag{1}$$

因此均值为公式(2)所示.

$$\bar{x}(t)=\frac{[x(t-m)+x(t-m+1)+...+x(t)+x(t+1)+...+x(t+m)]}{n} \tag{2}$$

其中, $n=2m+1$ 是均值滤波器的长度.

4.2 条形码图像灰度变换

对图像进行灰度变化, 其公式为公式(3):

$$\mathrm{Im}Gray=0.229*R+0.587*G+0.114*B \tag{3}$$

此公式是常规方法, 并且主要依赖 G 通道的图像值, 而在本文背景下, 灰度变化后的条形码区域可能会比较暗. 因此, 本文提出三颜色通道最大值灰度变换算法, 见公式(4):

$$\mathrm{Im}Gray(i,j)=\max\{imR(i,j),imG(i,j),imB(i,j)\}$$
$$1\le i\le m,1\le j\le n \tag{4}$$

其中: ImR(i,j), ImG(i,j), ImB(i,j)分别为 R、G、B 三色通道中坐标为(i,j)的像素值. m 和 n 分别为图像长和宽.

4.3 图像边缘处理

Sobel 边缘检测算子[8]的特点是计算简单、且对噪声有一定的平滑作用. 因此本文主要采用 Sobel 算子对条形码图像进行边缘化处理, Sobel 的基本思想是对图像灰度进行一阶求导, 进而判断导数是否连续来对边缘进行分割, 其卷积模板为:

$$Gx=\begin{bmatrix}-1&0&1\\-2&0&2\\-1&0&1\end{bmatrix}\quad Gy=\begin{bmatrix}-1&-2&-1\\0&0&0\\1&2&1\end{bmatrix}$$

Gx 及 Gy 分别代表经纵向向及横向边缘检测的图像, 图像的每一个像素的横向及纵向梯度近似值可用公式(5)表示, 来计算梯度的大小:

$$G=\sqrt{Gx^2+Gy^2} \tag{5}$$

然后可用公式(6)计算梯度方向:

$$q=\arctan(\frac{Gy}{Gx}) \tag{6}$$

Sobel 边缘检测算子是综合图像每个像素点四邻

域灰度值的加权和, 接近模板中心的权值较大. 适当选取阈值 T, 来判定是不是边缘点.

4.4 条形码图像二值化

对条形码进行二值化, 是为了剔除掉条形码边缘图像中的一些微弱的边缘信息, 消除对条形码边缘的影响. 在对条形码进行边缘化后, 选取一定的阈值对图像进行分割, 将图像的目标和背景按照灰度级划分为两个集合. 本文主要用到的阈值选取算法为 ostu 算法[9](最大类间方差法).

设图像像素数量为 N, 灰度范围为[0, K−1], 灰度值为 i 的像素个数为 n_i 个, 其出现的概率见式(7):

$$p_i = n_i / N \quad i = 0,1,2,...,K - 1 \tag{7}$$

其中: $\sum_{i=0}^{K-1} p_i = 1$

把图像中的像素按照灰度阈值 t 分为两类 A 和 B, A 由灰度值[0,t]之间的像素组成, B 由灰度值[t+1,K]之间的像素组成, 对于灰度的分布概率, 图像的整体均值见式(8):

$$u_T = \sum_{i=0}^{K-1} ip_i \tag{8}$$

则区域 A 和 B 的灰度均值分别见式(9)、(10):

$$u_A = \sum_{i=0}^{t} ip_i / w_0 \tag{9}$$

$$u_B = \sum_{i=t+1}^{K-1} ip_i / w_1 \tag{10}$$

其中: $w_0 = \sum_{i=0}^{t} p_i$, $w_1 = \sum_{i=t+1}^{K-1} p_i$, 由式(8)、(9)和(10)得:

$$u_T = w_0 u_A + w_1 u_B \tag{11}$$

两个区域 A、B 的方差见式(12):

$$\sigma^2 = w_0 (u_A - u_r)^2 + w_1 (u_B - u_T)^2 \tag{12}$$

按照最大类间方差的准则, 让 t 从 0 到 K−1 范围依次取值, 使 2σ 最大的 t 值即为最大类间方差的最佳阈值. Otsu 可以说是一种简单高效的方差法, 选取出来的阈值理想, 对各种情况的表现都较为良好, 是全局阈值自动选择的理想方法.

4.5 图像倾斜校正

由于条形码与相机不一定是保持理想的垂直, 故在进行图像处理的时候需要将倾斜的条形码图像加以矫正.

将图像的倾斜条形码校正, 则需要确定条形码的中心. 首先建立条形码的坐标系, 如图 3 所示.

图 3 倾斜图像坐标系

确定坐标系的中心点, 定义条形码像素与坐标的关系为 f(x,y), 则各像素在两坐标轴的最小值与最大值分别为式(13)、(14)、(15)、(16)所示.

$$x_{\min} = \min(x_{f(x,y)}) \tag{13}$$

$$x_{\max} = \max(x_{f(x,y)}) \tag{14}$$

$$y_{\min} = \min(y_{f(x,y)}) \tag{15}$$

$$y_{\max} = \max(x_{f(x,y)}) \tag{16}$$

式中 $x_{\min}$、$y_{\min}$ ——分别为像素点在系统坐标中 x, y 轴上的最小值;

$x_{\max}$、$y_{\max}$ ——分别为像素点在系统坐标中 x, y 轴上的最大值;

$x_{f(x,y)}$、$y_{f(x,y)}$ ——分别为像素点的横坐标和纵坐标.

可由式(17)计算中心坐标:

$$\begin{cases} x_{center} = (x_{\min} + x_{\max}) / 2 \\ y_{center} = (y_{\min} + y_{\max}) / 2 \end{cases} \tag{17}$$

由于条形码都是矩形, 其“条”的方向都一致且很规律, 因此由边缘化处理时所计算得到的梯度方向角即为条形码的倾斜角度, 由式(6)可得 θ.

将图像的中心移动到参考坐标原点, 则像素的中心坐标变为(0,0), 其他像素点的坐标变为(18):

$$\begin{cases} x' = x - x_{center} \\ y' = y - y_{center} \end{cases} \tag{18}$$

再将所有像素点绕着中心坐标, 即参考系的原点旋转一个 θ, 即式(19):

$$\begin{cases} x'' = x'\cos\theta - y'\sin\theta \\ y'' = x'\sin\theta + y'\cos\theta \end{cases} \tag{19}$$

通过上述计算, 即可实现对倾斜条形码的校正. 当一个 EAN-13 码被译出之后, 我们可以根据条形码第 13 码和其他 12 位译码之间的关系判断本次译码的准确性. 这里, 若我们定义奇性字符的和为 odd_sum, 偶性字符的和为 even_sum, 则如果满足式(20), 则说明译码准确:

$$d_{13}=[10-(odd\_sum+even\_sum*3)\%10] \quad (20)$$

5 实验结果

实验中通过接入的 USB 摄像头实时监测条形码, 并在拍摄到条形码图片时, 使用上述算法进行数据处理, 最终将所拍摄图像以及识别结果显示在 ARM 的 LCD 显示屏上, 图片上方的黑色数字即为所识别的结果, 如图 4 所示.

图 4 实验结果

在本嵌入式图像处理平台上, 图像处理系统可以以很高的实时性和准确率进行, 并且识别一副条形码所需时间平均为 300ms. 因此证明移植了 OpenCV 和 Qt 的嵌入式图像处理系统具有较高的稳定性和可行性.

参考文献

1 贾小军,喻擎苍.基于开源计算机视觉库 OpenCV 的图像处理.计算机应用与软件,2008,(4):276−278.
2 张静,叶梧,冯穗力.基于 ARM920T 的嵌入式 Linux 系统开发.现代电子技术,2005,4:22−24.
3 夏靖波,牛超.基于 S3C2440A 处理器的 Linux 移植的研究与实现.计算机与数字工程,2011,11:77−80.
4 吴燕燕,贺锋涛.基于 ARM9 平台上 Qt/Embedded 的移植与开发.液晶与显示,2013,2:261−265.
5 马智,叶林,葛俊锋.ARM9+Linux 平台上计算机视觉的实现.计算机与数字工程,2011,12:134−137,150.
6 彭远斌,丛新元,杨民.食品溯源系统的物品编码设计.现代农业科技,2014,1:295−297.
7 王科俊,熊新炎,任桢.高效均值滤波算法.计算机应用研究,2010,2:434−438.
8 陈含,吕行军,田凤珍,董倩,王克俭,韩宪忠.基于 Sobel 算子边缘检测的麦穗图像分割.农机化研究,2013,3:33−36.
9 覃晓,元昌安,邓育林,石亚冰,元建.一种改进的 Ostu 图像分割法.山西大学学报(自然科学版),2013,4:530−534.